ガスタービンエンジン制御用の
マルチコアマイコン向けコード生成

トヨタ自動車株式会社
未来創生センター　S-フロンティア部
第2エネルギーグループ
池内　祥人
mail: yoshito_ikeuchi@mail.toyota.co.jp

Frontier Research Center
TOYOTA

アジェンダ

1. 部署・プロジェクト紹介
2. 制御系開発の対象
3. 開発手法
4. 並列処理設計事例
5. 実装・検証
6. まとめ

アジェンダ

1. 部署・プロジェクト紹介
2. 制御系開発の対象
3. 開発手法
4. 並列処理設計事例
5. 実装・検証
6. まとめ

トヨタのガスタービン開発

1964　1970　1980　1990　2000　2010

トヨタ自動車
東富士研究所

Preliminary Study

GT Hybrid

GTV

産業用GT開発

(所属部署)S-フロンティア部

GT関連技術の研究開発

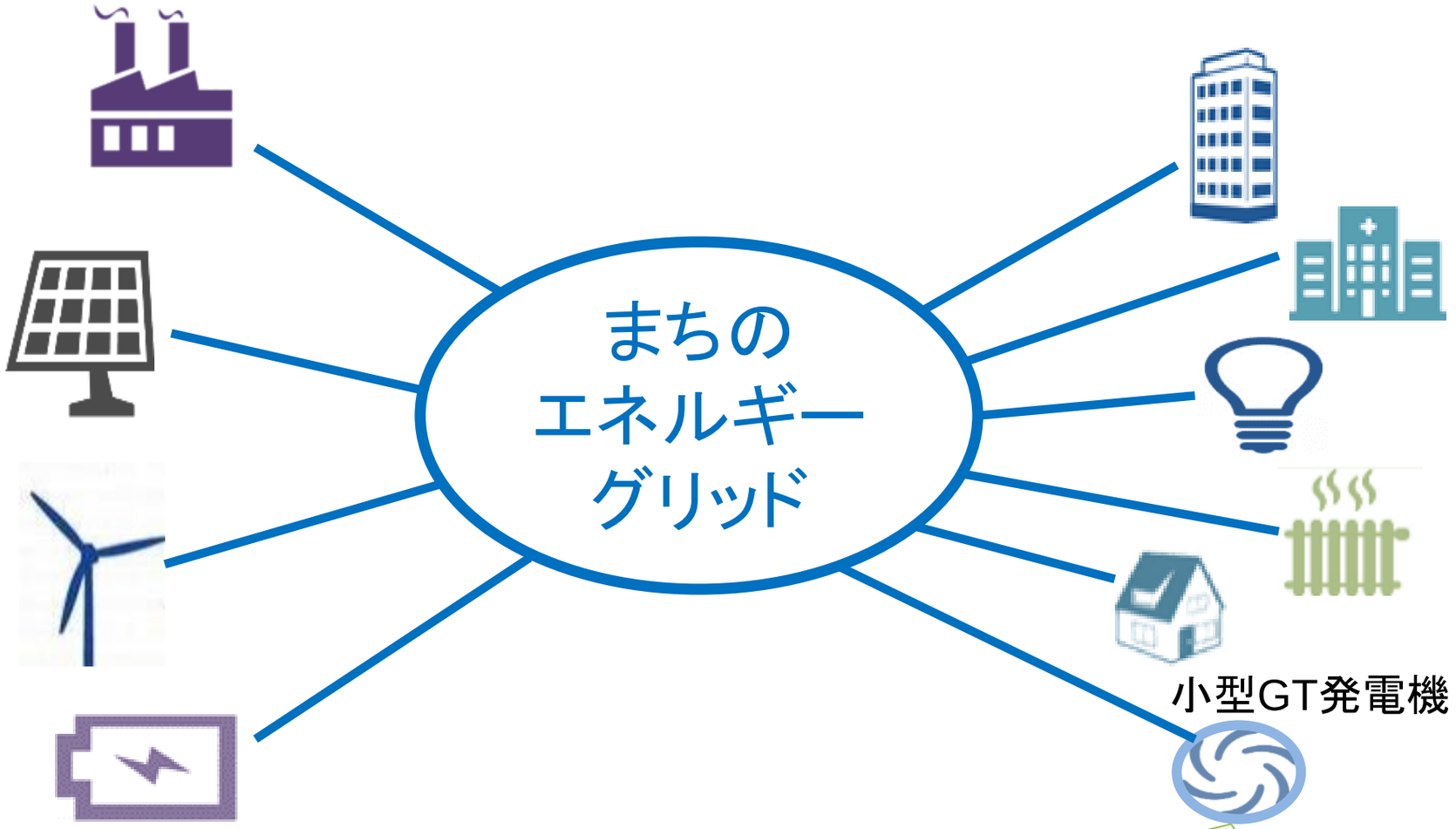
産業用マイクロGT発電機から事業化

’98年トヨタタービンアンドシステム
→’18トヨタエナジーソリューションズ

マイクロGT
コジェネシステム販売

アンモニア燃焼GT
SOFC向けGT
開発

分散型エネルギー社会へ向けた小型GT発電機開発プロジェクト



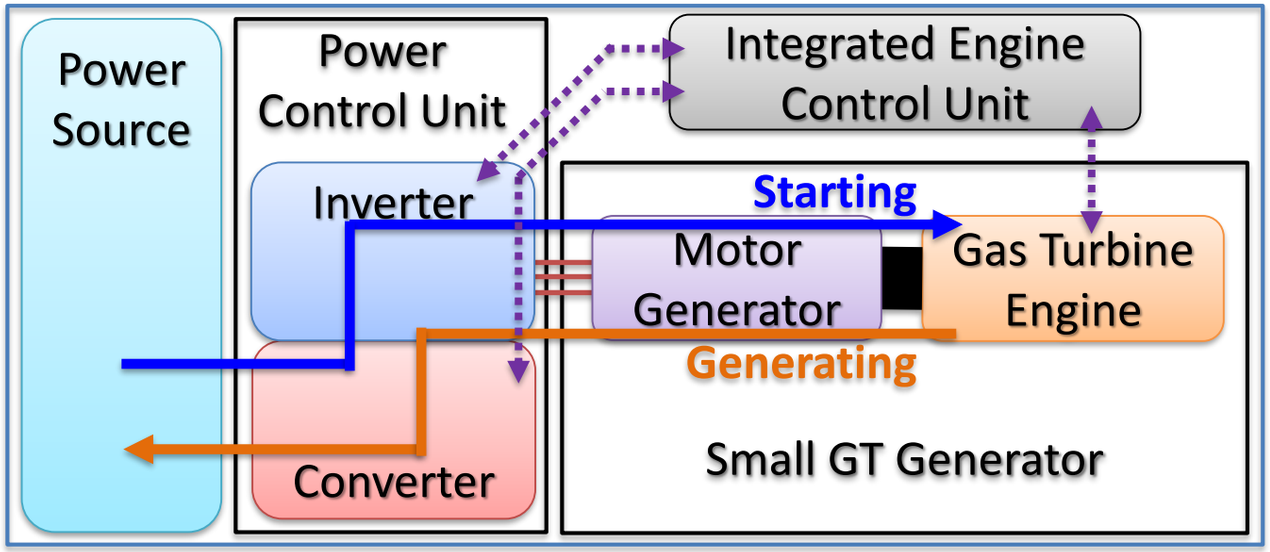
複雑化するエネルギーシステムに対応できる組込制御の開発
<u>マルチコアマイコンへの実装を前提とした、モデルベース開発事例をご紹介</u>

アジェンダ

1. 部署・プロジェクト紹介
2. 制御系開発の対象
3. 開発手法
4. 並列処理設計事例
5. 実装・検証
6. まとめ

分散電源用ガスタービン発電機の制御系開発

ガスタービン発電機の構成



本制御系開発のポイント

①シングルコアマイコンの処理性能限界に到達
- ルネサスエレクトロニクス社製モータ制御マイコン”RH850/C1H(2コア)”を採用
- 開発初期からマルチコアマイコンへの実装を想定

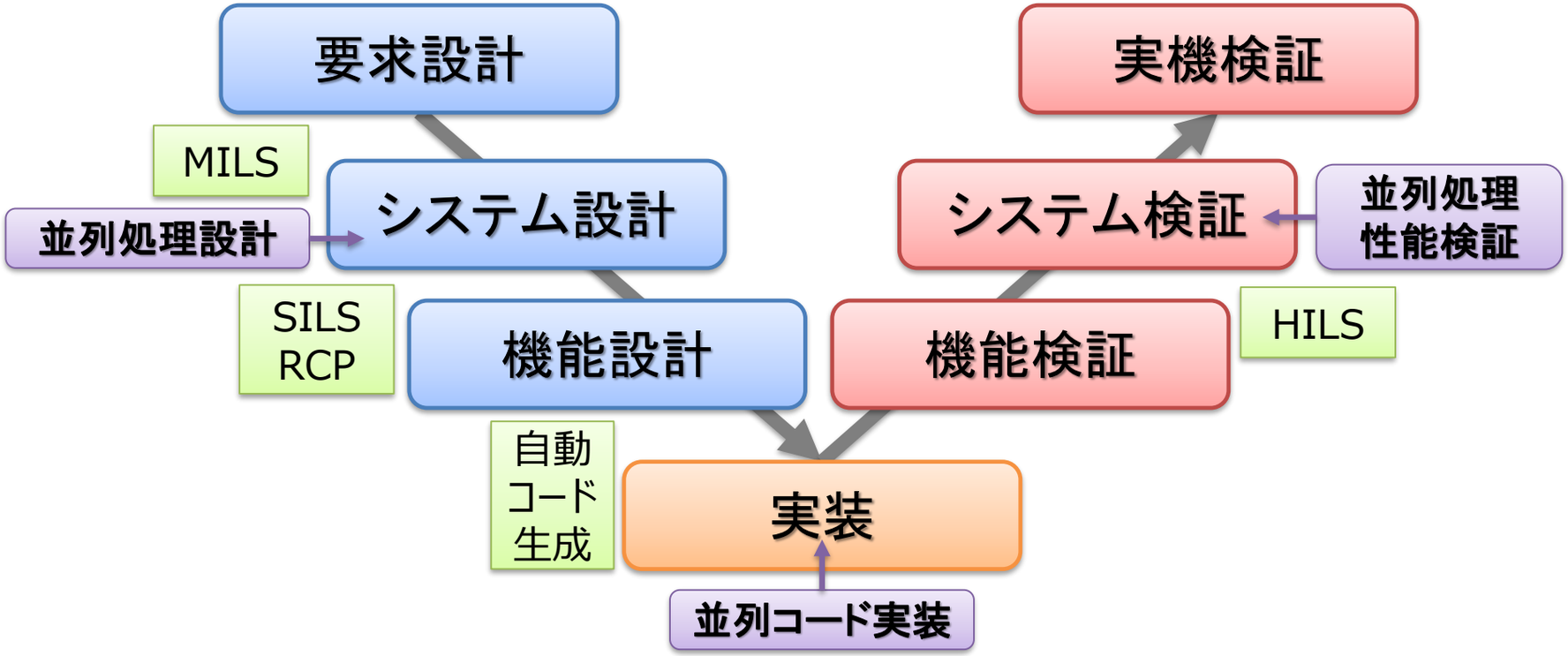
②モデル設計から実機検証まで、全プロセスをモデルベースで新規開発
- 従来環境の制約なく、新手法導入が容易

マルチコアPILS導入による開発効率化事例をご紹介

アジェンダ

1. 部署・プロジェクト紹介
2. 制御系開発の対象
3. 開発手法
4. 並列処理設計事例
5. 実装・検証
6. まとめ

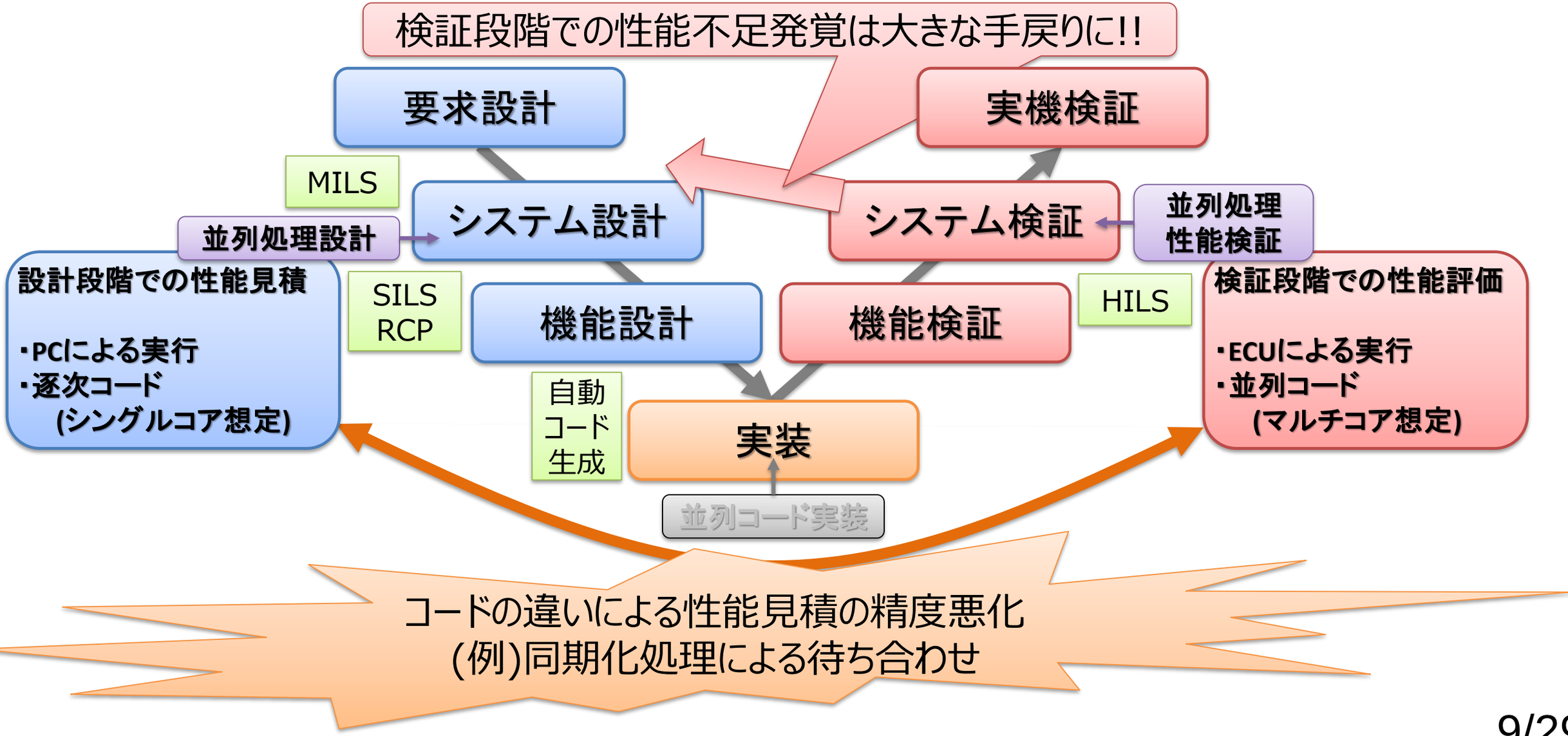
マルチコア実装を想定した開発プロセスの課題



設計・実装・検証の各段階で並列化に伴う作業が追加
従来のモデルベース開発手法の課題は?

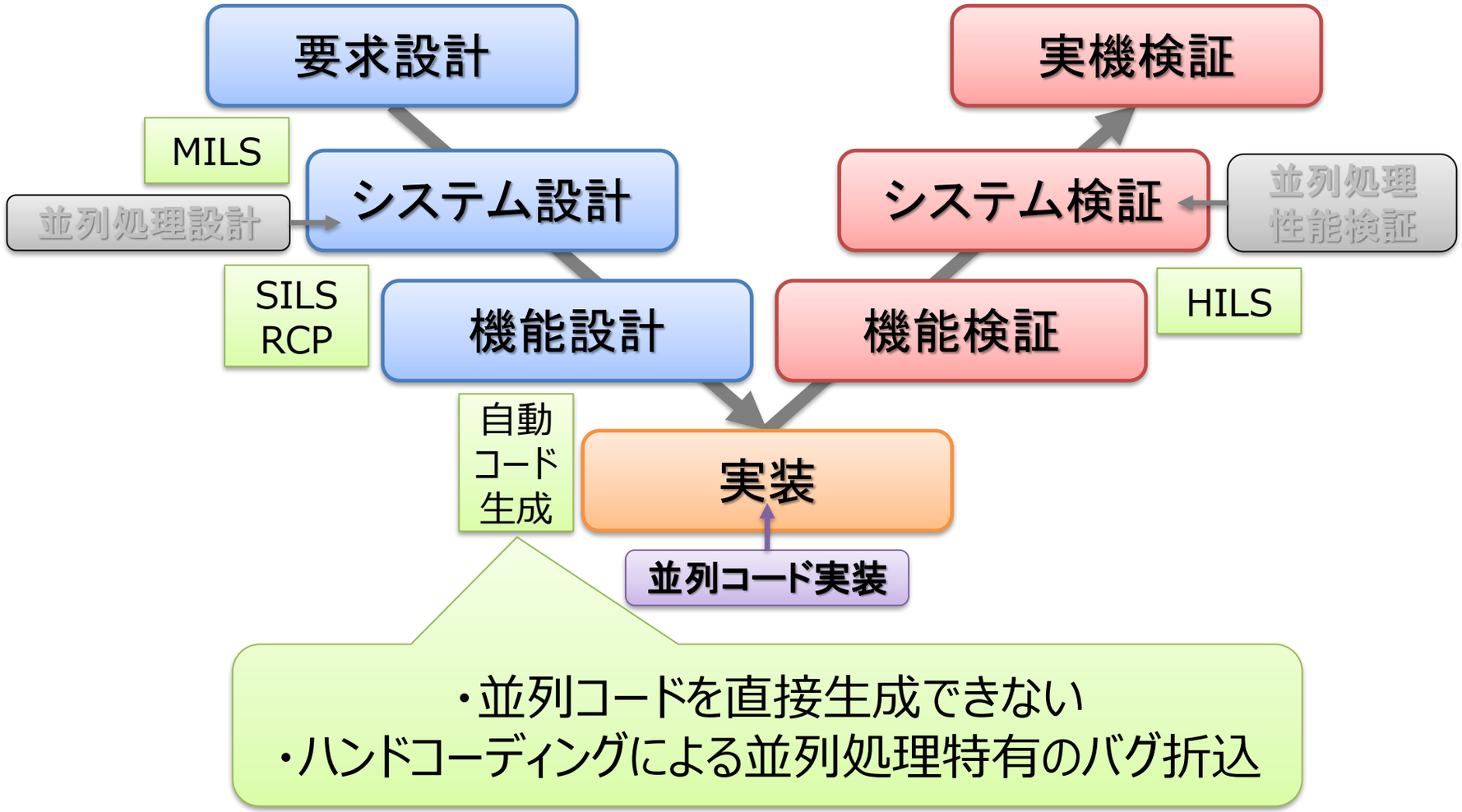
マルチコア実装を想定した開発プロセスの課題

【課題①】並列処理性能の見積



マルチコア実装を想定した開発プロセスの課題

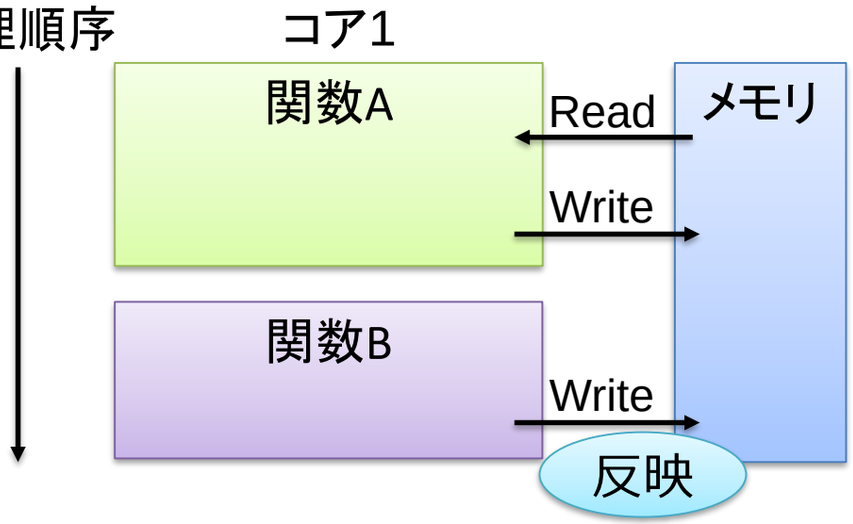
【課題②】並列コードの実装



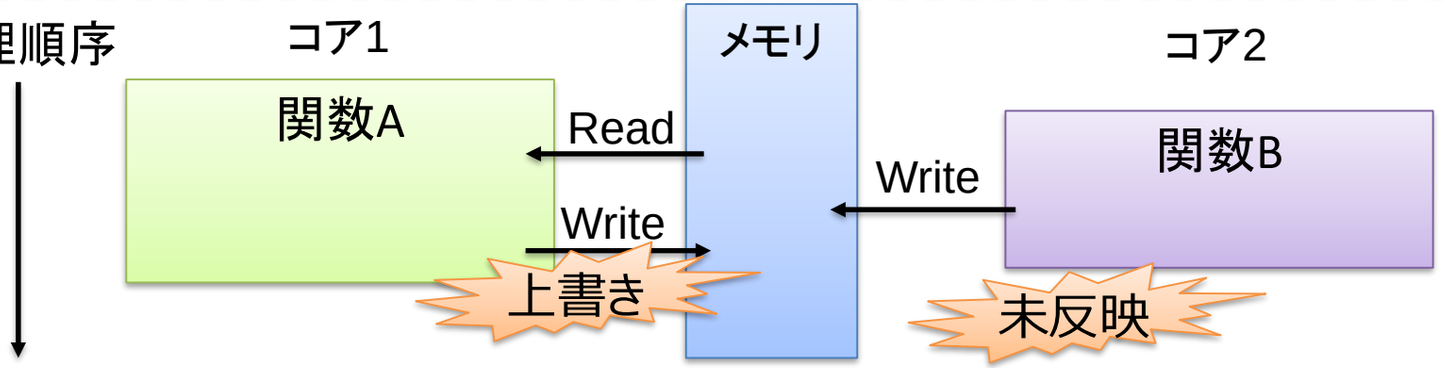
並列処理特有のバグ

メモリのアクセスパターン変化による意図せぬ動作

・**シングルコアの場合**

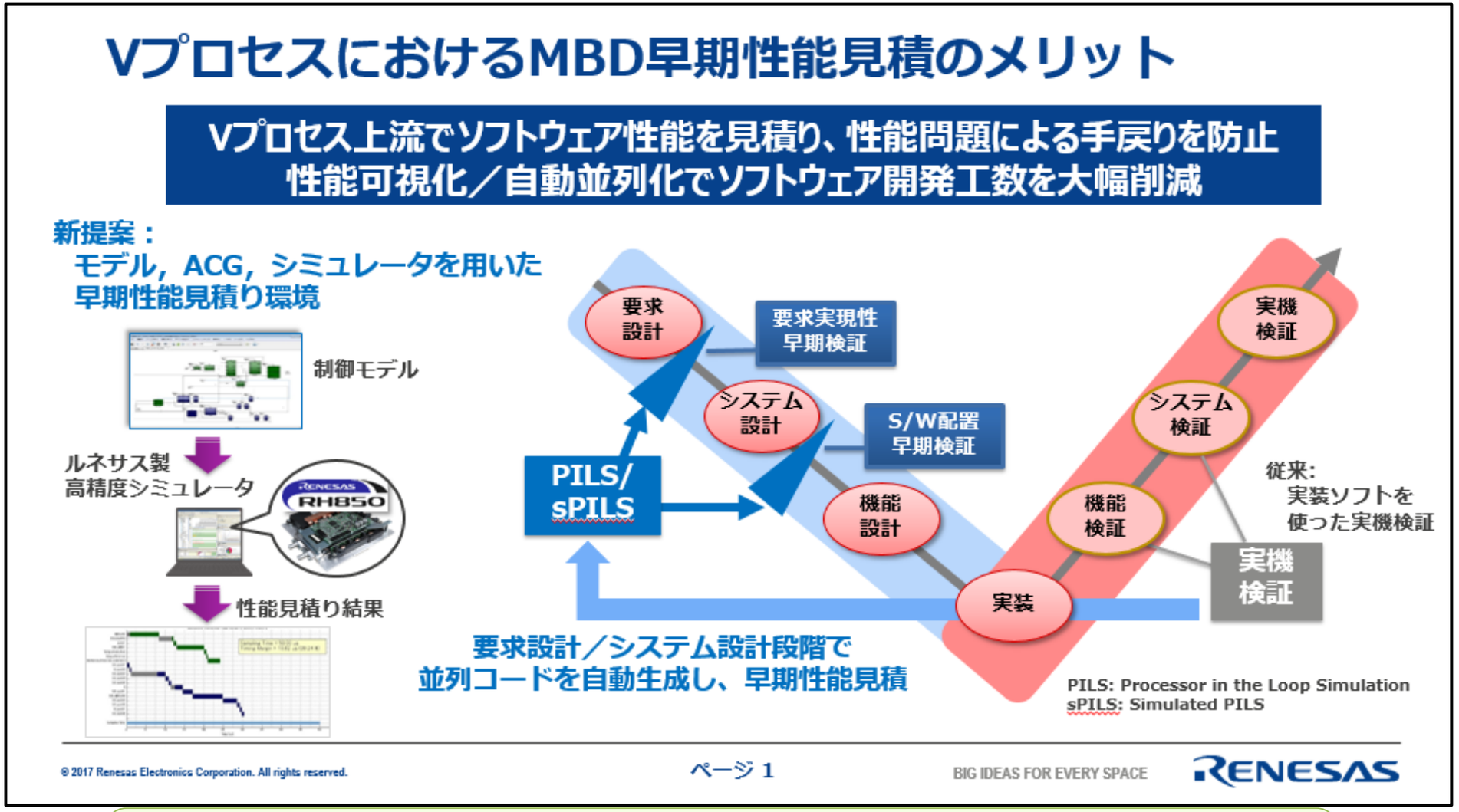


・**マルチコアの場合**



並列処理化によりメモリアクセス順序を適切に設定しないと
意図しない反映・未反映が発生

ルネサスエレクトロニクス様が提唱するマルチコアPILSによる早期性能見積

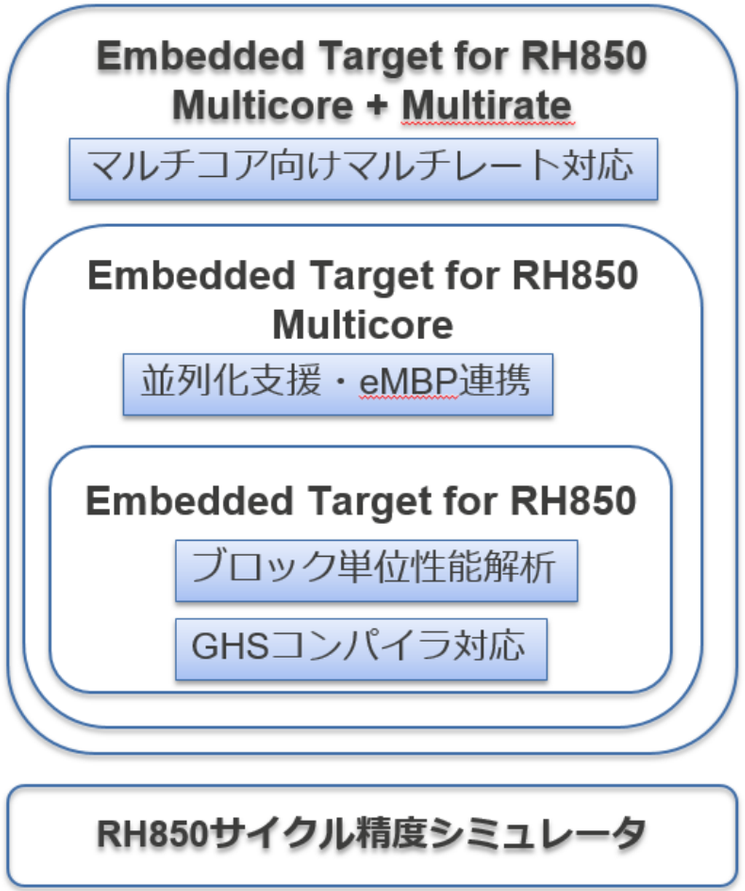


ルネサス
エレクトロニクス様より
許諾得て掲載

マルチコアマイコンへの実装を想定した場合
マルチコアPILSは強力な開発支援ツールとなりうる

導入したマルチコアPILS環境

RH850モデルベース開発環境



マルチコア・マルチレート・シングルタスクに対応

導入ソフト

- **Embedded Target for RH850 Multicore + Multirate**
  - 複数の制御周期(**マルチレート**)対応
  - 以下の機能を含む

- **Embedded Target for RH850 Multicore**
  - イーソル製モデルベース並列化ツールeMBPと連携し **Simulinkモデルの並列化**を支援
  - 以下の機能を含む

並列化コア数・タスク数が多くないため
自動並列化ツールは導入せず

- **Embedded Target for RH850**
  - シングルコア向け（並列化支援なし）
  - マルチレート対応
  - **ブロック単位**性能解析、 GHSコンパイラ対応

実機コントローラ無しでも
PILS検証をするために導入

導入ソフト

- **RH850サイクル精度シミュレータ**
  - Embedded Targetと連携して性能評価、動作検証が可能
  - PC環境だけで実機に近い精度の時間計測が可能

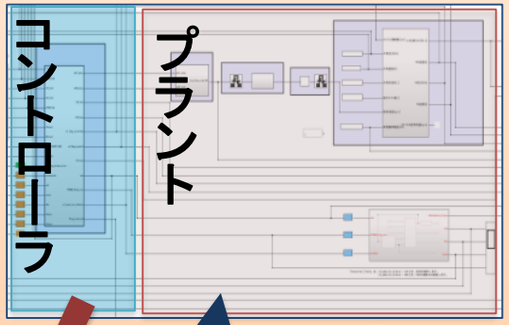
Page 1

BIG IDEAS FOR EVERY SPACE RENESAS

ルネサス
エレクトロニクス様より
許諾得て掲載

マルチコアPILS環境

MILS
(MATLAB®/Simulink®)

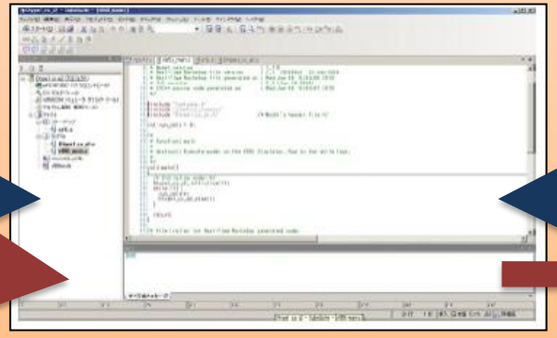


仮想環境
(PC)

実機コントローラ
(RH850マイコン)

プラント挙動と
コントローラ処理結果
(処理時間含む)
を連成解析

統合開発環境(CS+)



デバッグ動作

コンパイル
&仮実装

エミュレータ(E1)

①コントローラ仮実装

②PILS検証

PILS: Processor In the Loop Simulation

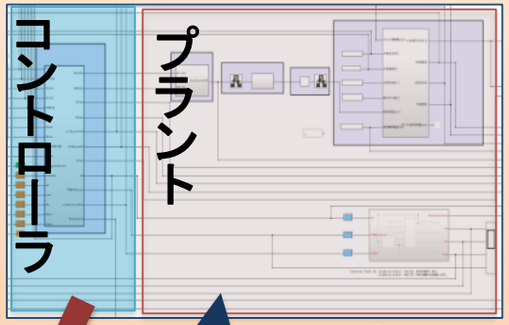
コントローラモデルから
並列コードを自動生成

MILSモデルがあれば実プロセッサによる
正確な並列処理時間、演算結果の検証が可能
(ただし、モデル演算部のみ)

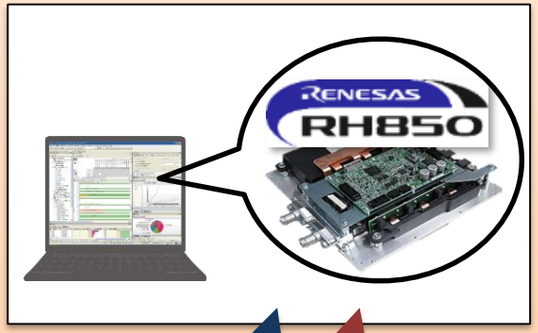
サイクル精度シミュレータを用いたsPILS

仮想環境
(PC)

MILS
(MATLAB®/Simulink®)

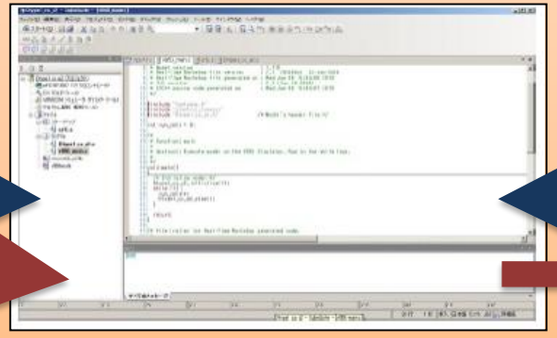


仮想コントローラ
(サイクル精度シミュレータ)



プラント挙動と
コントローラ処理結果
(処理時間含む)
を連成解析

統合開発環境(CS+)



デバッグ動作

コンパイル
&仮実装

①コントローラ仮実装

②PILS検証

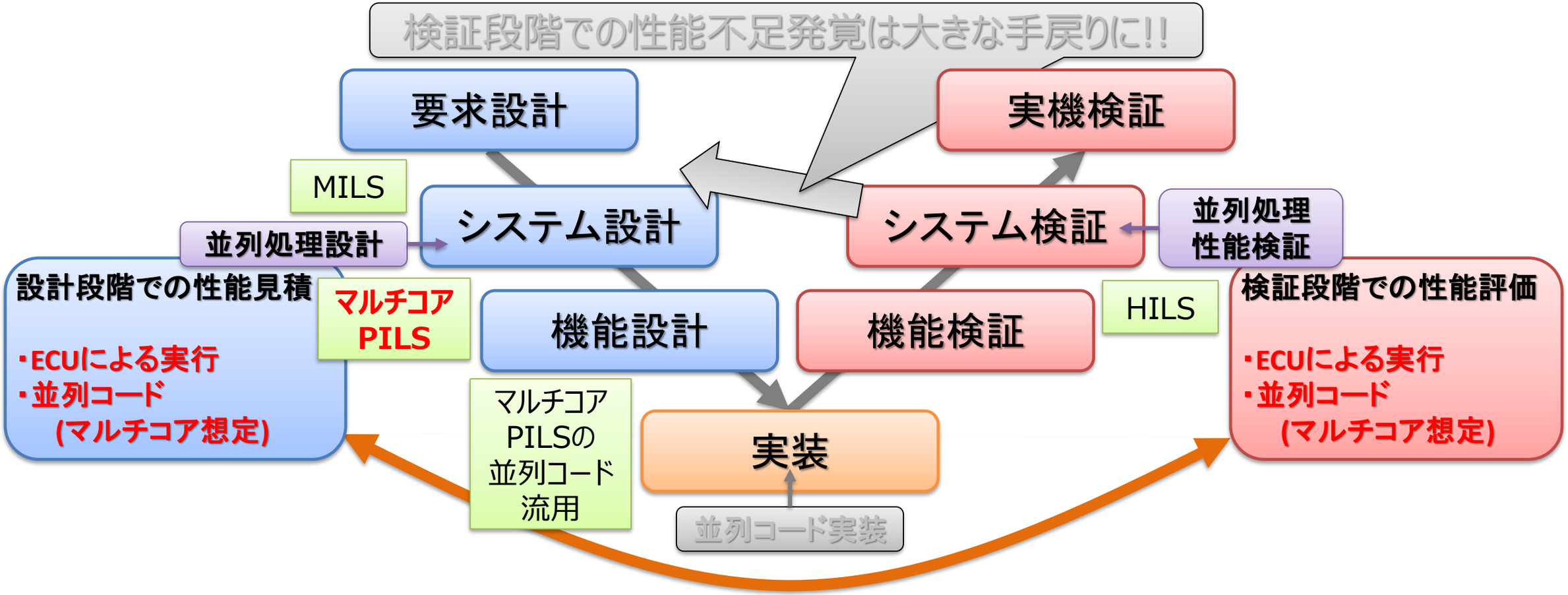
sPILS: simulated PILS

コントローラモデルから
並列コードを自動生成

サイクル精度シミュレータを用いることで
実機コントローラが無くてもPILS検証可能

マルチコアPILS導入による開発課題の解消

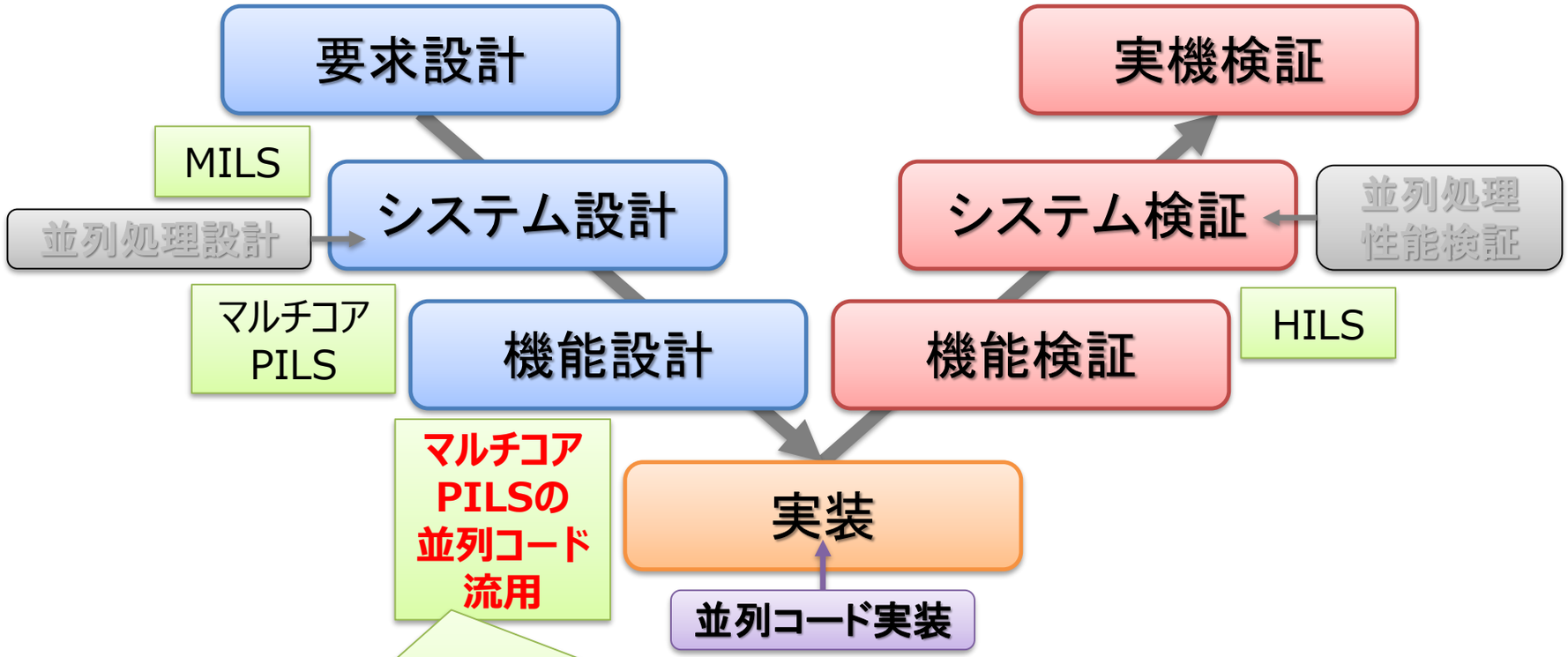
【課題①】並列処理性能の見積



・設計段階で実プロセッサ・並列コードによる検証が可能になり性能見積の精度が向上
・検証過程がフロントローディング化でき、HILS工程の簡略化にも貢献

マルチコアPILS導入による開発課題の解消

【課題②】並列コードの実装



マルチコアPILS検証時に生成される並列コードを流用
→手作業による並列化を削減、バグ排除&実装効率化

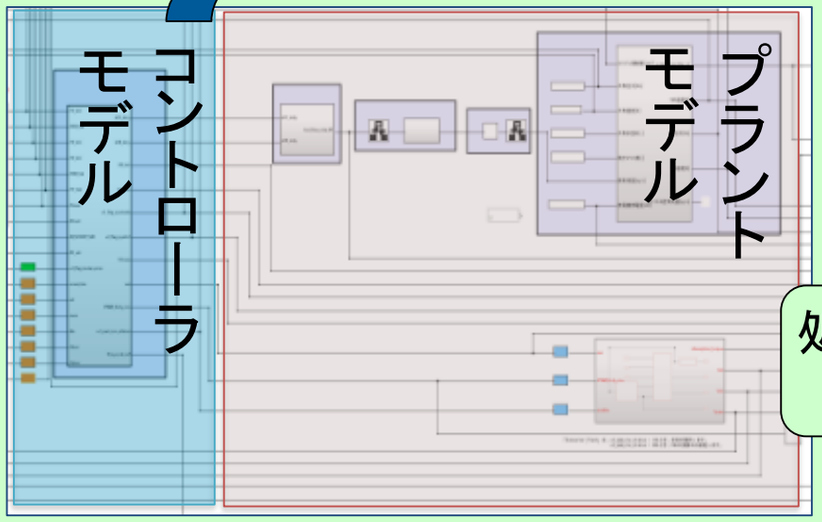
アジェンダ

1. 部署・プロジェクト紹介
2. 制御系開発の対象
3. 開発手法
4. 並列処理設計事例
5. 実装・検証
6. まとめ

マルチコアPILSによる並列処理設計

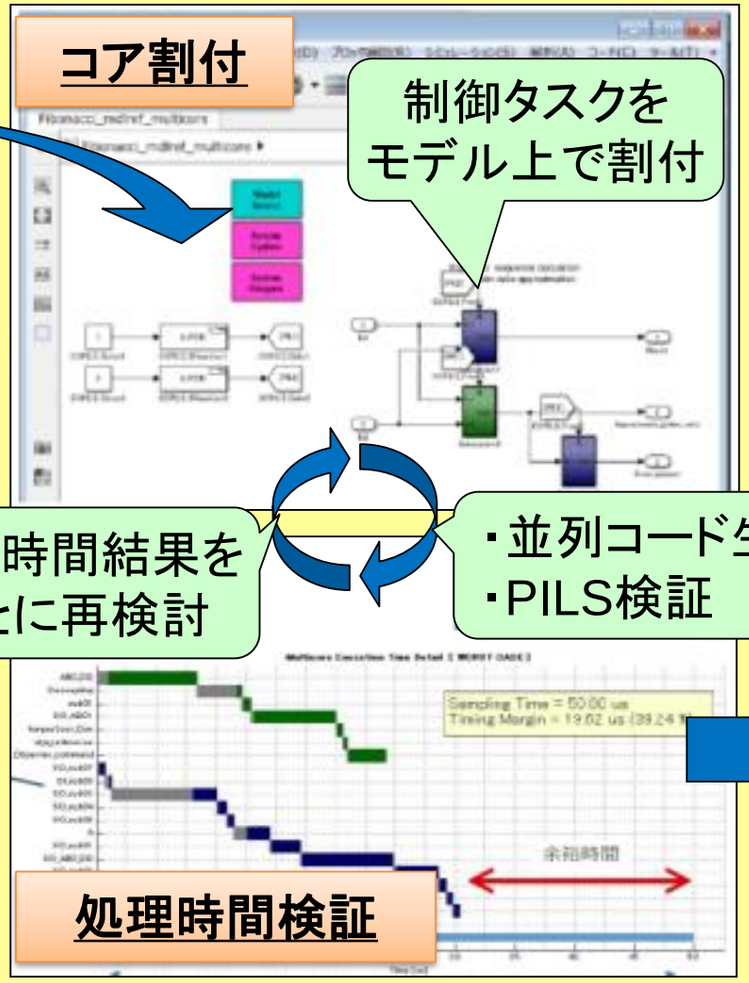
マルチコアPILS

MILS



コア割付

制御タスクを
モデル上で割付



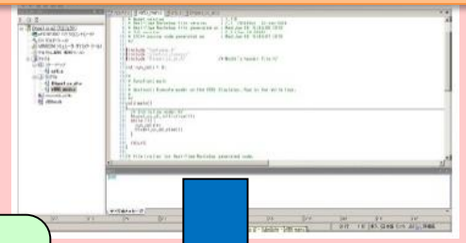
処理時間結果を
もとに再検討

・並列コード生成
・PILS検証

処理時間検証

本実装

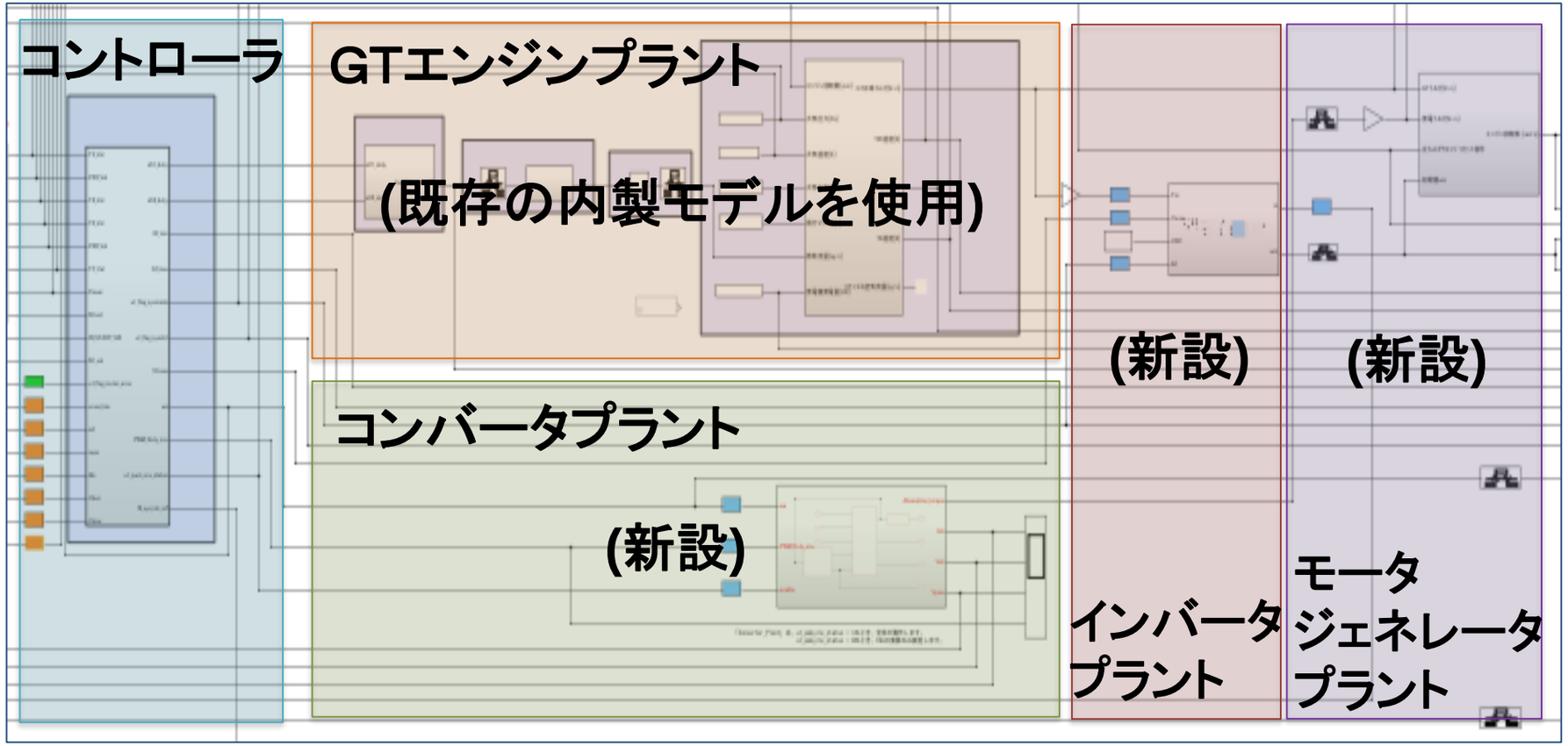
**ミドルウェアコード構築
(ハンドコーディング)**



【本実装】
並列生成コードと
ミドルウェアコードを
結合、ECUへ実装

【並列処理設計】
コア割付&処理時間検証のサイクルを回し
設計段階で処理性能目処づけ

MILS構築



パワエレ系のプラントモデルを新設
→Simscape Electrical™による低コスト・早期立ち上げ

使用製品

・MATLAB®
・Simulink®

【コントローラ構築】
・Control System Toolbox™
・Simscape™

【プラント構築】
・Simscape Electrical™
(旧SimPowerSystems™)

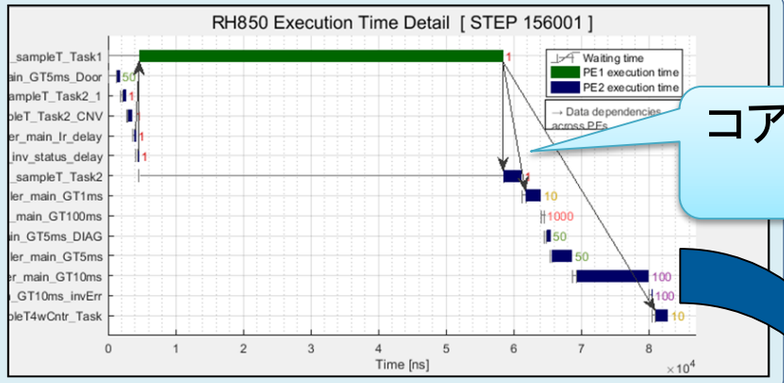
【コード生成】
・Embedded Coder®

マルチコアPILSによる並列化検討

検討初期

最悪処理時間：81μs
(シングルコアと同等の動作)

PE1
100μs Task
→
PE2
1ms Task

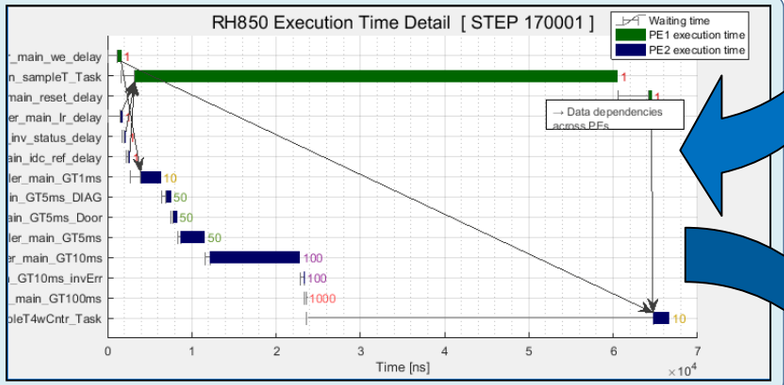


コア1の演算結果をそのまま
コア2の演算に使用

検討中期

最悪処理時間：67μs

PE1
100μs Task
→
$\frac{1}{z}$
Unit_Delay
→
PE2
1ms Task



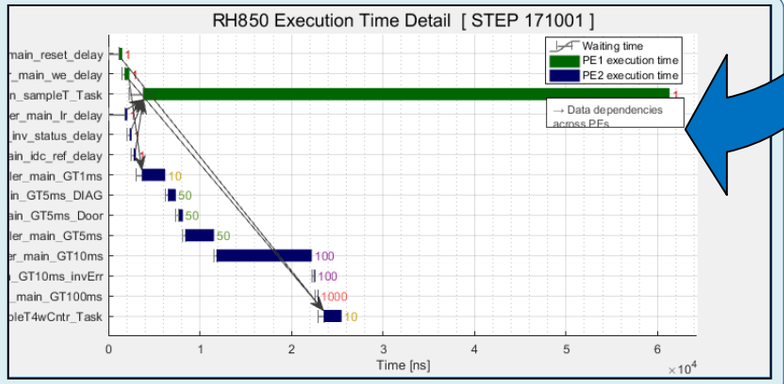
【コントローラモデルの修正】
コア2の演算に
コア1演算結果の前回値を使用
(制御機能への影響も検証可)

検討後期

最悪処理時間：61μs

優先順位②
優先順位①

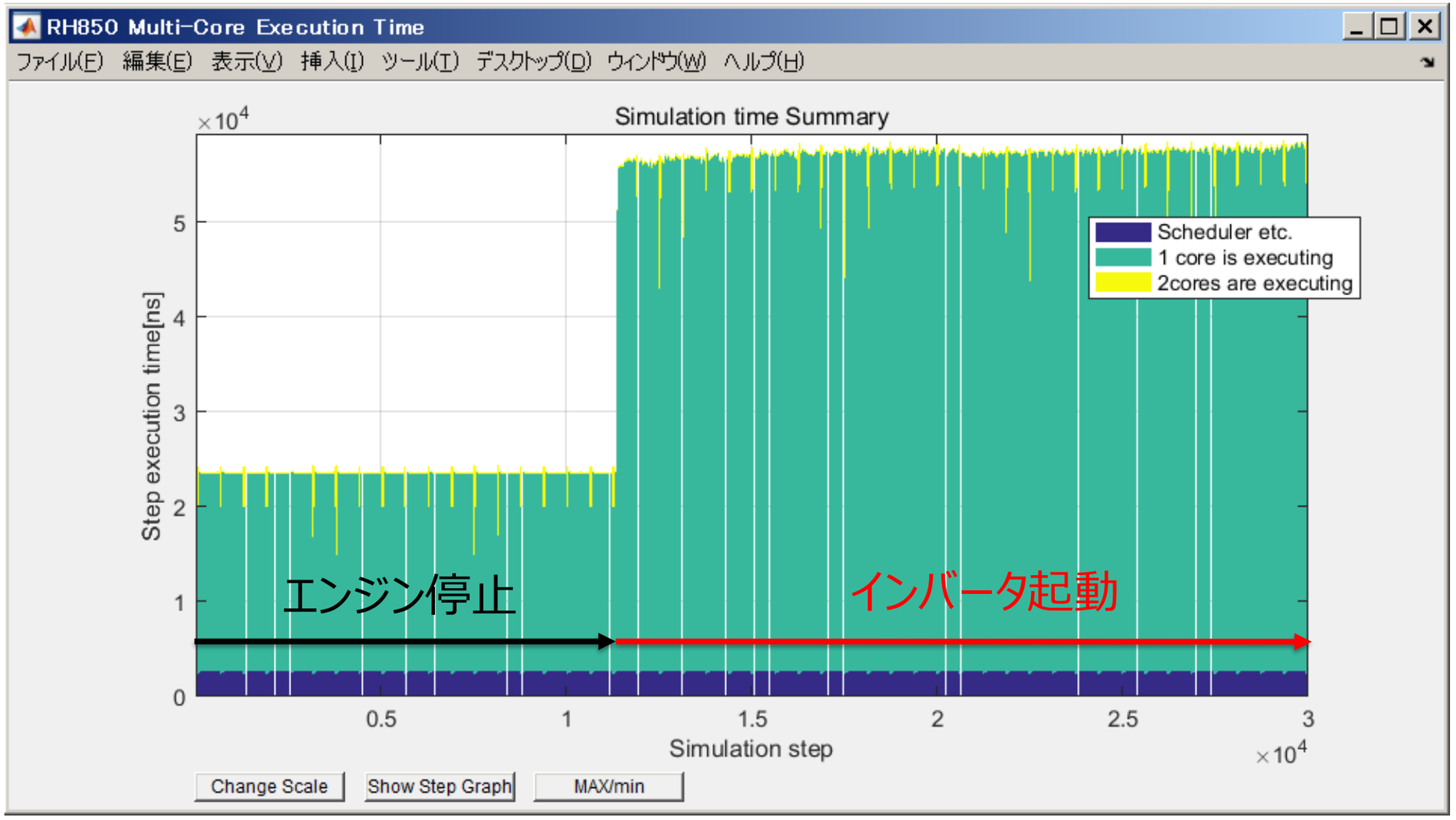
PE1
100μs Task
→
$\frac{1}{z}$
Unit_Delay
→
PE2
1ms Task



【優先順位の調整】
サブシステム単位で
実行順序を見直し、並列性向上

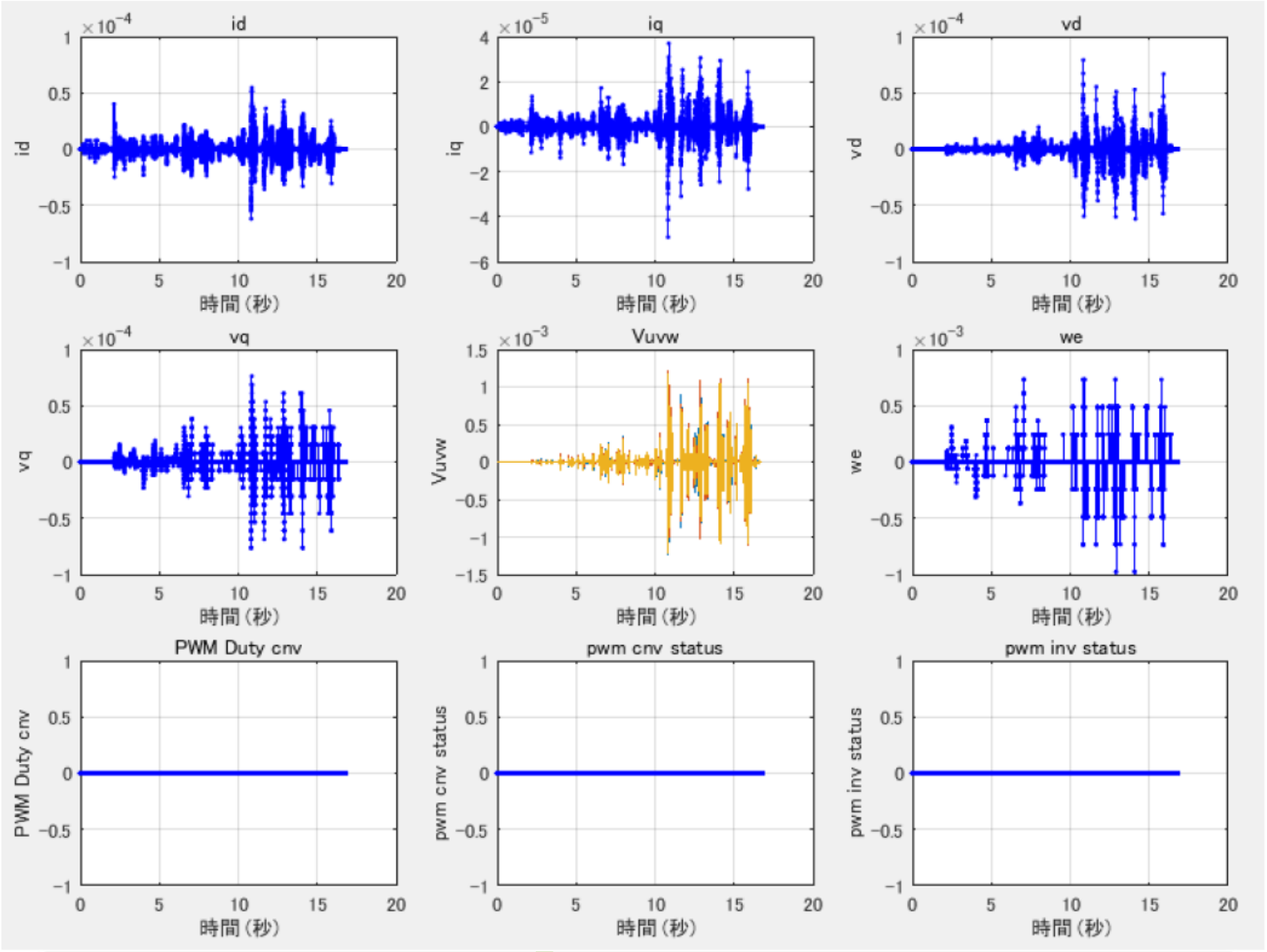
最悪処理時間を指標に
PILS検証サイクルを回し、
優れた並列処理設計を実現

検証シナリオを通した処理時間の確認



検証における全ステップの処理時間分析が可能

MILS-マルチコアPILSの検証結果比較



MILSとマルチコアPILSを同時に実行し、
コントローラ演算結果の差異を確認

アジェンダ

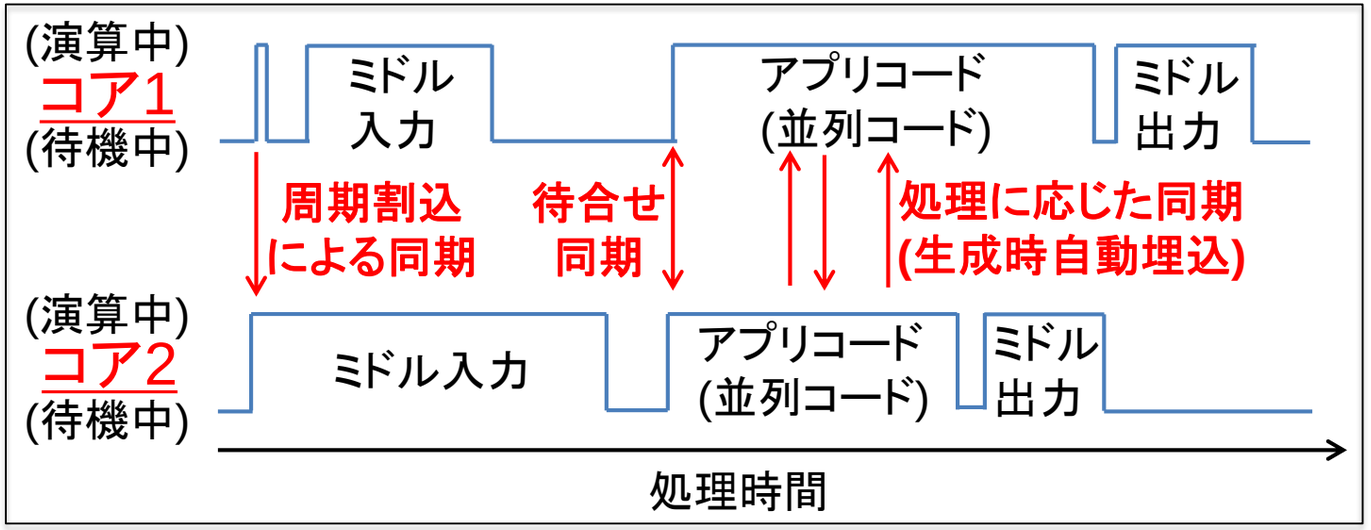
1. 部署・プロジェクト紹介
2. 制御系開発の対象
3. 開発手法
4. 並列処理設計事例
5. 実装・検証
6. まとめ

マルチコアPILSの並列コードを流用した実装

生成される並列コードの特徴

①マルチコア・マルチレート・シングルタスク方式
②最小処理周期を基本周期とした関数(タスクスケジュール機能含む)
③同一コードで各コア処理に対応
④同期化処理コードの自動埋込(ルネサスエレクトロニクス社製ライブラリ:libipcx)

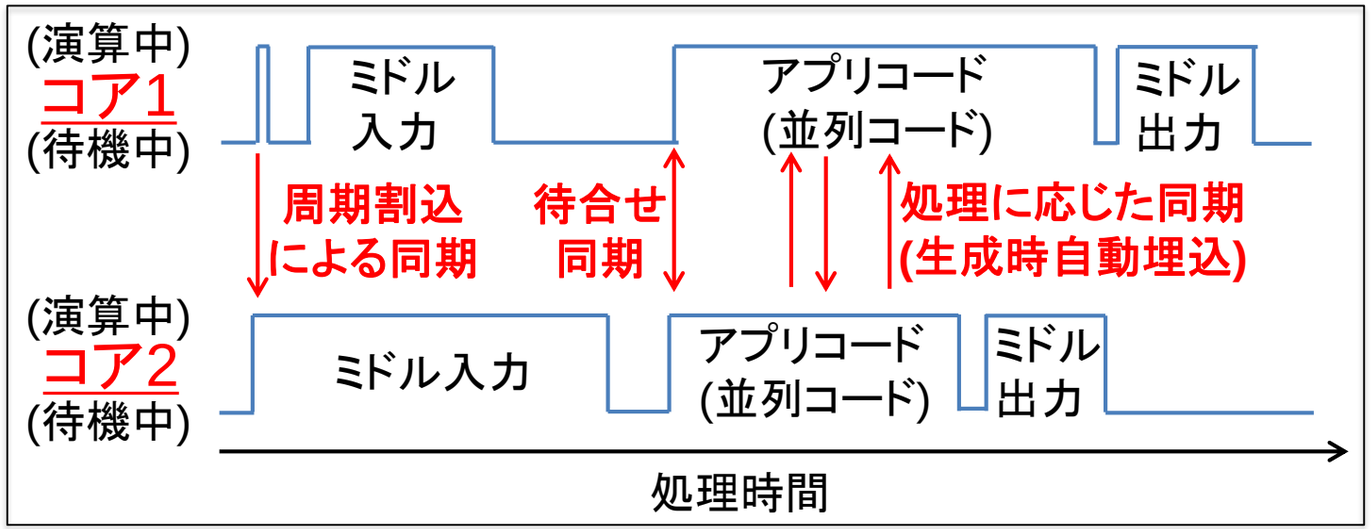
各コアのアプリコードとしてそのまま流用可能



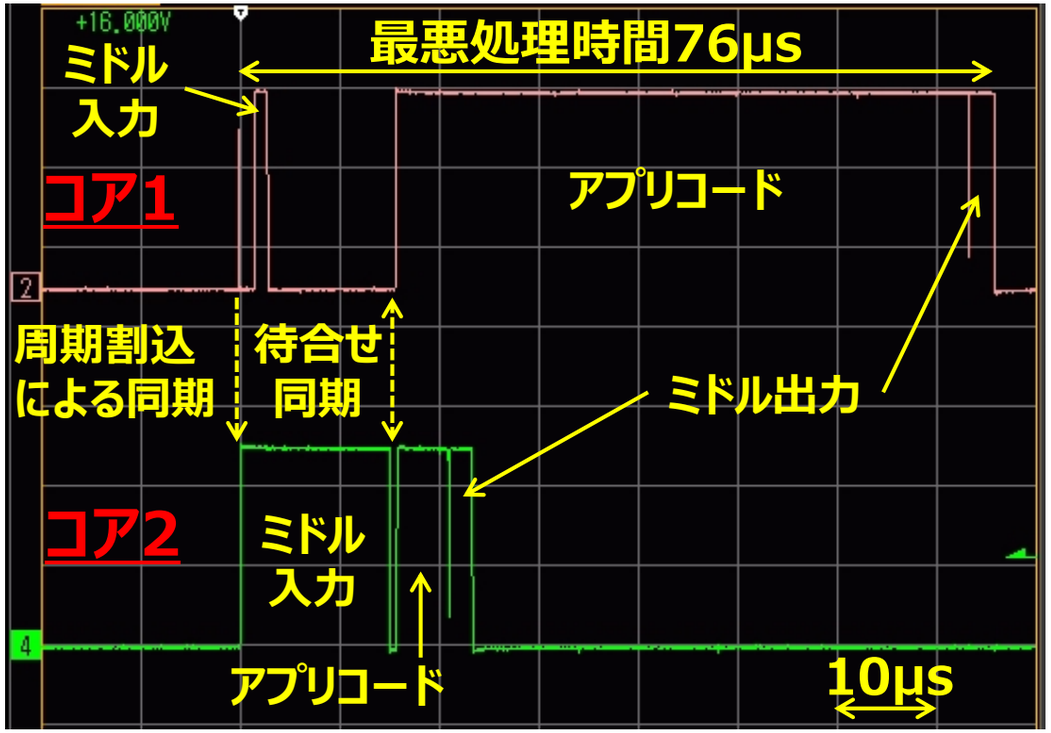
ミドルコード結合後の演算処理イメージ

但し、本実装においてもアプリコード先頭の同期処理が必要

本実装による処理時間検証



ミドルコード結合後の演算処理イメージ



HILS検証時の演算処理結果

ミドルウェア処理の偏り、見積が課題
→Simulink上にモデル化することで、PILSによる処理時間評価が可能

アジェンダ

1. 部署・プロジェクト紹介
2. 制御系開発の対象
3. 開発手法
4. 並列処理設計事例
5. 実装・検証
6. まとめ

まとめ

マルチコアマイコンへの実装を想定した制御系開発において
マルチコアPILS環境を導入、処理時間を考慮した並列処理設計を実施

これにより、下記開発効率化を実現

①並列化検討サイクルの高速化
②検証段階における処理性能不足の回避(手戻りゼロ)
③生成された並列コード流用による実装効率化
④実機検証(HILS)の簡略化

今後メニーコア化による処理時間見積の困難化が予想され、
処理時間を考慮した並列処理設計が重要となる

マルチコアPILSは上記課題を解決し得る有効な開発支援ツールであり、
今後の発展、普及が期待される

マルチコアPILSの課題

①処理時間見積の精度向上

マルチコアPILSの処理時間検証はモデル表現部のみ

【解決策】：①ハンドコード処理部のモデル化

※モータ制御IPモデル、ドライバソフトは
ルネサスエレクトロニクス様より販売中

②並列コードを流用した実装支援

ユーザーにとっては、本実装&実機動作が重要

【要望】：①本実装ガイドライン
②マクロ等による実装条件分岐

③モデル制約の削減

マルチコアPILS適用には、一部モデル(使用ブロック・構造等)に制約がある

【要望】：①ソフトの改善
②チェックツール
③ガイドライン等の拡充

※ルネサスエレクトロニクス社製
マルチコアPILSに関するお問い合わせ
https://www.renesas.com/jp/ja/support/contact.html

Frontier Research Center

TOYOTA

TOYOTA